TEKNILLINEN TIEDEKUNTA

Älykkäät teollisuusrobotit

Ville Lauronen

Prosessitekniikan tutkinto-ohjelma

Kandidaatintutkielma

Maaliskuu 2021

TIIVISTELMÄ

Älykkäät teollisuusrobotit

Ville Lauronen

Oulun yliopisto, Prosessitekniikan tutkinto-ohjelma

Kandidaatintyö 2021, 36 s.

Työn ohjaaja yliopistolla: Jukka Hiltunen

Työssä tutustutaan teollisuusrobotteihin, ja niihin elementteihin, joiden avulla robotit saadaan toimimaan älykkäästi. Tavoitteena on ymmärtää, kuinka älykäs teollisuusrobotti toimii, ja mitä asioita sen toimintaan liittyy. Tässä kirjallisuuskatsauksessa tiivistyy perustietous älykkäistä teollisuusroboteista siten, että lukija kykenee ymmärtämään, mitä asioita hänen tulee osata ja tietää, kun tavoitteena on implementoida älykäs robotti teollisuuden sovelluskohteeseen. Työhön on koottu perustietoutta muun muassa teollisuusrobottien mekaniikasta ja ohjelmoimisesta, sekä simuloinnista.

Tutkielmassa esitellään robotiikan tärkeimmät termit ja teollisuusrobottityypit, sekä tutustaan robottien hallitsemiseen tarvittavaan matematiikkaan, sisältäen perustietoutta manipulaattoreiden kinematiikasta, dynamiikasta ja singulariteeteista. Ohjelmointiosuudessa käydään lävitse teollisuusrobottien ohjelmointi- ja opetustapoja, sekä esitellään uusia, lähitulevaisuuden ohjelmointimetodeja. Lopuksi tutustutaan erilaisiin simulointiympäristöihin, kokeillaan muutamaa simulaatiota, ja pyritään ymmärtämään ROS-ympäristön tuomia hyötyjä älykkäiden robottien kehityksessä. Tavoitteena on ymmärtää, miten älykkäitä robotteja ohjelmoidaan, ja mitä asioita tulee tietää ja hallita, jotta älykäs robotti saadaan implementoitua teollisuuden sovellukseen.

Asiasanat: teollisuusrobotti, älykäs teollisuusrobotiikka, manipulaattori, robotin ohjelmointi, robottisimulaatio

ABSTRACT

Intelligent Industrial Robotics

Ville Lauronen

University of Oulu, Degree Programme in Process Engineering

Bachelor's thesis 2021, 36 pp.

Supervisor at the university: Jukka Hiltunen

The purpose of this thesis was to introduce the elements, needed for making industrial robots intelligent, and provide the basic knowledge of industrial robots. The work is a literature review, explaining what basic knowledge is needed for implementing an intelligent robot for an industrial application. The scope includes the basic knowledge of mechanics, programing, mathematical modeling, and controlling the robots.

The thesis introduces the fundamentals of industrial robotics, covering the different manipulator types and the basic mathematics, including the manipulator kinematics, dynamics, and singularities. At the programming section, the basics of robotic programming is discussed, covering the programming languages, teaching methods and the near future trends. At the end, the different simulation environments are explored, and the significance of ROS is recognized. The goal is to understand, how the intelligent industrial robots could be programmed and simulated, and what knowledge is needed at the implementation of intelligent industrial robots for the industrial application.

Keywords: industrial robot, intelligent industrial robotics, manipulator, robot programming, robotic simulator

SISÄLLYSLUETTELO

1 JOHDANTO

Uudet, älykkäämmät ja entistä ketterämmin ohjelmoitavat teollisuusrobotit soveltuvat myös uudenlaisiin tehtäviin, joissa robotteja ei ole aiemmin kyetty hyödyntämään. Yleiskäyttöisistä, tuotannon muutoksiin nopeasti mukautuvista roboteista on haaveiltu aina, mutta robottien ohjelmoimisen hitaus on ollut iso haaste, joten teollisuusrobotteja on totuttu näkemään lähinnä volyymifokusoituneilla aloilla.

Uusin teknologia mahdollistaa entistä nopeamman asiakkaiden tarpeisiin reagoimisen, mikä tuo suuria hyötyjä myös perinteisille toimijoille. Kaikki tuotanto ei ole volyymikeskeistä, ja ketteryys on kasvava kilpailuvaltti myös volyymifokusoituneella aloilla. Tuotantolinjan nopea konfigurointi uusille tuotteille säästää aikaa ja rahaa, ja mahdollistaa entistä paremman palvelun tarjoamaisen asiakkaille.

Uusien robottien entistä nopeampi ohjelmoitavuus, helppous ja mukautuvuus ovat keskeisiä syitä robottien nopeaan yleistymiseen. Perinteisen teollisuusrobotin toiminta nojaa joukkoon yksinkertaisia liikesarjasekvenssejä, joita robotti suorittaa peräjälkeen. Tässä työssä tutkitaan, kuinka robotti saadaan toimimaan älykkäämmin, esimerkiksi keinoälypohjaisia menetelmiä hyödyntäen. Samalla esitellään myös tapoja, joilla robotteja voidaan ohjelmoida entistä nopeammin.

Älykkäät robotit kykenevät hyödyntämään keinoälyä vähintäänkin jonkin yksittäisen osatehtävän toteutuksessa. Käytännössä tämä voi tarkoittaa esimerkiksi sitä, että robotti käyttää konenäköä selvittämään, miten kappaleesta kannattaa ottaa kiinni. Älykkäät robotit kykenevät generoimaan liikeratansa automaattisesti sensoreilta saamansa datan avulla, aivan kuten myöhemmin esitellyssä simulointidemossa tullaan näkemään.

Työn tavoitteena on selvittää, kuinka älykäs teollisuusrobotti toimii, mitä asioita sen toimintaan liittyy ja miten älykkäitä robotteja ohjelmoidaan. Tämän työn luettuaan lukija kykenee ymmärtämään, mitä perusasioita hänen tulee osata ja tietää, kun tavoitteena on implementoida älykäs robotti teollisuuden sovellukseen.

2 ÄLYKÄS ROBOTIIKKA

Keinoälyltä on aina odotettu paljon, mutta se on antanut odottaa itseään. Historiassa on nähty suuria lupauksia ja suuria pettymyksiä. Uusimmat analyysit kuitenkin osoittavat, että keinoäly on vihdoin ja viimein alkanut kantaa hedelmää. (Bughin, Hazan, *ym.*, 2017) Tässä työssä tutustutaan teollisuusrobotiikkaan ja keinoälyn hyödyntämiseen teollisuusrobottien ohjauksessa, sekä pohditaan uusia älykkäitä sovelluksia ja niiden tuomia hyötyjä.

Robottien käyttö teollisuudessa on lisääntynyt voimakkaasti viime vuosina. Vuonna 2019 teollisuusrobotteja oli käytössä 2,7 miljoonaa kappaletta, mikä on lähes kolminkertainen määrä vuoteen 2009 verrattuna. Robottien määrän kasvun on ennustettu jatkuvan kovana, kunhan COVID-19-pandemian aiheuttamasta lamasta päästään ylitse. (International Federation of Robotics, 2020)

Robottien määrän kasvu nojaa omalta osaltaan robottien ohjaus- ja ohjelmointimenetelmien kehittymiseen. Teollisuusrobotit ovat muuttumassa entistä älykkäämmiksi, mikä monipuolistaa robottien käyttömahdollisuuksia. Uusia ketterästi ohjelmoitavia teollisuusrobotteja ja cobotteja voidaan hyödyntää myös sellaisissa käyttökohteissa, joihin robotit eivät ole aiemmin soveltuneet.

Yhteistyörobotit, eli niin sanotut cobotit (engl. c*ollaborative robots, cooperative robots, cobots* or *robot assistants*) ovat loistava esimerkki uuden älykkään teknologian tuomista mahdollisuuksista. Yhteistyörobotit kykenevät toimimaan samassa tilassa ihmisen kanssa, ja auttamaan esimerkiksi kokoonpanotehtävää suorittavaa ihmistä. Cobottien kaltainen uusi teknologia tuo kuitenkin myös haasteita. Esimerkiksi yhteistyörobotit eivät vaadi samanlaista turva-aitausta, kuin perinteiset teollisuusrobotit, joten turvastandardit täytyy suunnitella uusiksi. Toisaalta yhteistyörobottien vaatimattomammat turvalaitevaatimukset tuovat myös kustannussäästöjä, ja tehostavat tilankäyttöä. (Vysocky ja Novak, 2016)

2.1 Historia

Sana ”robotti” on tiedettävästi käytetty ensimmäistä kertaa 1921 ilmestyneessä tšekinkielisessä näytelmässä R.U.R (Rossum’s Universal Robots), jossa robotit kuvataan ihmismäisiksi koneiksi, jotka työskentelevät väsymättä kellosta piittaamatta. (Murray, Li, *ym.*, 1994, s. 1) Kuvaus sopii sangen hyvin nykyaikaisiin teollisuusrobotteihin, mutta automaatiolaitteen ja robotin välinen raja on sangen häilyvä. Milloin kyseessä on robotti, ja milloin kyse on vain automaattisesti toimivasta koneesta? Kansainvälinen standardi määrittelee robotin seuraavasti: ”Robotti on ohjelmoitava mekaaninen laite, jolla on vähintään kaksi liikettä, joiden avulla se kykenee toteuttamaan sille tarkoitettuja tehtäviä automaattisesti.” (*ISO 8373:2012*, 2012) Robotin määritelmä puree siis sellaisiinkin laitteisiin, joita emme ole tottuneet pitämään robotteina.

Automaattisesti toimivia robotteja on käytetty teollisuudessa ainakin vuodesta 1961 lähtien, kun ensimmäinen Unimation inc:n ohjelmoitava robotti asennettiin General Motorsin tehtaalle. Kokemukset ensimmäisestä teollisuusrobotista olivat hyviä, ja pian muutkin yhtiöt alkoivat kehittämään teollisuusrobotteja. (Murray *ym.*, 1994, s. 3)

Sittemmin teollisuusrobotit ovat levinneet laajempaan käyttöön, sillä erityisesti tietotekniikan kehitys on mahdollistanut entistä luotettavampien, monipuolisempien ja helpommin ohjelmoitavien robottien kehittämisen.

2.2 Robotiikka ja keinoäly

Perinteinen robottien ohjelmoiminen on haastavaa ja hidasta puuhaa, sillä jokainen robotin tekemä liikerata määritellään erikseen koodaamalla, tai opettamalla. Perinteinen tyhmä robotti noudattaa siis vain koodarin ohjeita, kykenemättä muuhun. Älykkäät robotit kykenevät hyödyntämään keinoälyä vähintäänkin jonkin yksittäisen osatehtävän toteutuksessa. Käytännössä tämä voi tarkoittaa esimerkiksi sitä, että robotti käyttää konenäköä selvittämään, miten kappaleesta kannattaa ottaa kiinni, minkä pohjalta robotti generoi liikeratansa automaattisesti.

Koneäly, keinoäly ja tekoäly ovat kaikki samaa asiaa tarkoittavia trendisanoja, joita käytetään useissa eri asiayhteyksissä. Keinoälyn määritteleminen ei ole helppoa, sillä edes alan kirjallisuus ei kykene esittämään sille yksiselitteistä määritelmää. Joskus yksinkertaisiakin päättelymekanismeja kutsutaan keinoälyksi, mutta toisaalla samoja menetelmiä pidetään vain tavallisena koodina tai tilastomatematiikkana. Ei ole olemassa yleisesti hyväksyttyä teoriaa, joka selittäisi mitä äly on. Yhden määritelmän mukaan keinoäly onkin asia, jonka määritelmää ja toteutusta ei vielä täysin ymmärretä. (Pietikäinen ja Silven, 2019, s. 17)

Yleisesti voidaan kuitenkin todeta, että keinoälyllä viitataan niihin menetelmiin, joilla koneet saadaan toimimaan ihmisen älyn kaltaisesti ja rationaalisesti. Robotiikan saralla tämä tarkoittaa sitä, että koneet kykenevät toimimaan automaattisesti ilman täsmällisiä ohjeita, perustuen esimerkiksi erilaisilta sensoreilta saatuun dataan.

Erityisesti konenäön alalla on otettu suuria harppauksia 2010-luvulla. Tämä viimeaikainen kehitys on perustunut erityisesti syväoppiviin neuroverkkoihin, joiden kehitys on ollut huimaa. (Pietikäinen ja Silven, 2019, s. 6) Teollisuusrobotiikka on eräs niistä sovelluskohteista, joissa konenäön kehittymisestä on suurta hyötyä. Esimerkiksi useat edellä mainitun yhteistyörobotiikan sovellutukset voivat perustua konenäköön, joka kykenee seuraamaan ihmisen sijaintia (Maragkos, Vosniakos, *ym.*, 2019).

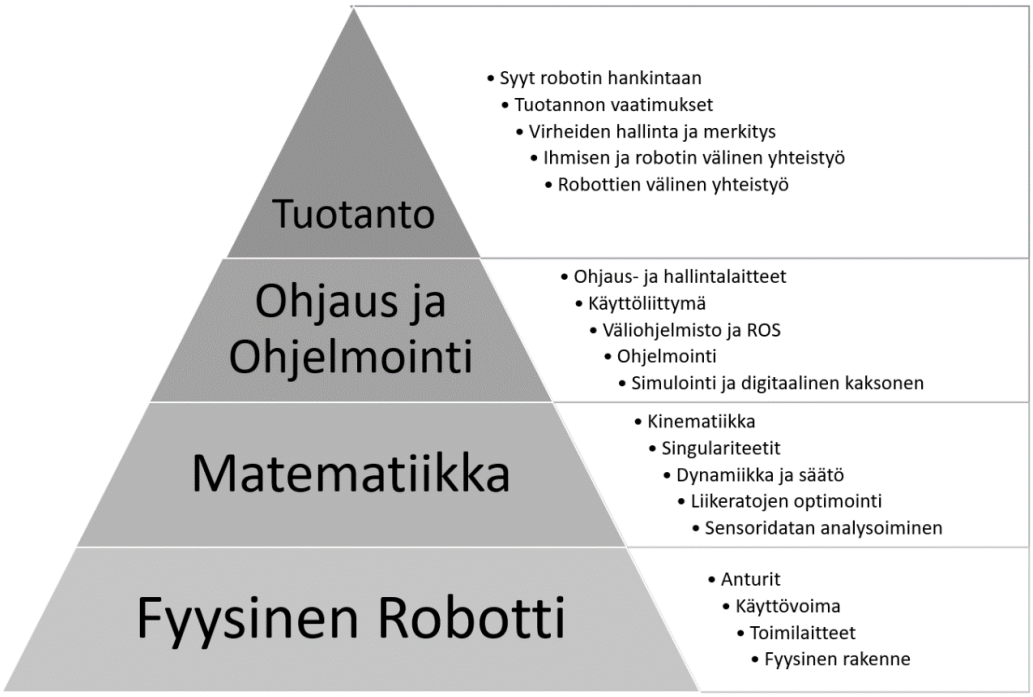
Konenäköä voidaan hyödyntää esimerkiksi tartuttavan kappaleen identifioimisessa, jonka jälkeen robotin tulisi kyetä suunnittelemaan omat liikkeensä siten, että nivelrobotti kykenee ottamaan kiinni kappaleesta. Keinoälyä voidaankin hyödyntää myös robotin liikkeiden suunnitelelussa ja optimoinnissa.

Oleellista on huomata, että älykkään robotin tekeminen on *hankalaa*. Se vaatii runsaasti poikkitieteellistä osaamista. Ymmärrystä vaaditaan muun muassa teollisuudesta, tuotantotekniikasta, mekaniikasta, matematiikasta ja tietotekniikasta, sekä myös ohjelmoinnista. Näistä enemmän seuraavassa kappaleessa.

3 TEOLLISUUSROBOTTI

Kansainvälisen standardisoimisjärjestön (*International Organization for Standardization,* ISO) määritelmän mukaan teollisuusrobotti on automaattisesti ohjattu, uudelleen ohjelmoitava ja monikäyttöinen manipulaattori, jossa on vähintäänkin kolme liikettä. Manipulaattori, on useista liikuteltavista varsista koostuva mekanismi, jolla voidaan liikutella objekteja, kuten työkaluja. Teollisuusrobotti on monikäyttöinen ja uudelleenohjelmoitava laite, mikä tarkoittaa sitä, että samaa fyysistä alustaa voidaan käyttää useissa eri teollisuuden tehtävissä. Teollisuusrobotti voi olla kiinteästi paikalleen asennettu, tai mobiilisti liikkuva laite. (*ISO 8373:2012*, 2012)

Kirjallisuudessa keskitytään yleensä vain tiettyihin robotin ominaisuuksiin tai toimintoihin, mutta tässä osiossa pyrin luomaan kokonaiskuvan robotin toiminnasta. Tavoitteena on ymmärtää robottia oliona, jonka toimintaa voidaan kuvata usealta eri tasolta, useasta eri näkökulmasta.



Kuva 1. Näkökulmia teollisuusrobotiikkaan.

3.1 Mekaniikka –Robotti on mekaaninen kone

3.1.1 Fyysinen rakenne

Tavallisesti teollisuusrobotti koostuu jäykistä kappaleista, jotka on kytketty toisiinsa liikkuvilla mekanismeilla, kuten lineaarisesti liikkuvilla varsilla ja pyörivillä nivelillä. (Murray *ym.*, 1994, s. 81) Manipulaattorin tilaa, eli konfiguraatiota voidaan muuttaa toimilaitteiden, eli *aktuaattoreiden* avulla, jolloin manipulaattori saadaan liikkumaan hallitusti.

Toimilaite voi olla tyypiltään esimerkiksi hydraulisesti liikuteltava varsi, tai niveltä pyörittävä sähkömoottori. Toimilaitteen käyttövoimana voi toimia hydrauliikka, pneumatiikka tai sähkökäyttö. (Murray *ym.*, 1994, s. 155) Käyttövoiman valinta on riippuvainen esimerkiksi robotin tyypistä, toimintaympäristöstä, sekä voimalle, tarkkuudelle ja nopeudelle asetetuista vaatimuksista.

Työkalu (engl. *End Effector*) on manipulaattorin päähän liitettävä osa, jota robotti hyödyntää tehtävänsä suorittamiseen (*ISO/TR 20218-1:2018*, 2018). Työkalu voi olla esimerkiksi tarttuja, maaliruisku tai hitsauslaite. Työkalu voi olla myös vaihdettava, jopa siten, että robotti vaihtaa työkaluaan automaattisesti eri työvaiheiden välissä.

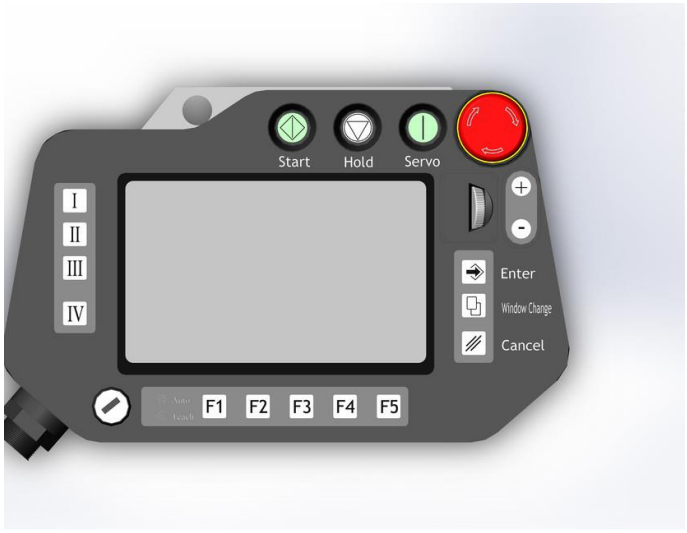
Työkalun efektiivisen osan sijainti, eli TCP (engl. *Tool Center Point*) on robotin ohjelmoinnin kannalta oleellinen muuttuja (*ABB Robotics Operating manual: RobotStudio 5.13*, 2010, s. 22). Se voi kuvata esimerkiksi kappaleenkäsittelyrobotin tarttujan koordinaatteja, joiden hallittuun liikuttamiseen robotin toiminta perustuu.

3.1.2 Ohjaus- ja hallintalaitteet

Ohjelmoitava logiikka, eli PLC (engl. *Programmable Logic Controller*) on laite, joka vastaanottaa anturisignaaleja, käsittelee niitä ohjelmointinsa mukaisesti ja ohjaa erilaisia teollisuuden laitteita ja manipulaattoreita.

Jokaisella teollisuusrobotilla on ohjausyksikkö (engl. *robot controller*), joka on robotin ohjaamiseen erikoistunut PLC. Se on robotin osa, joka lukee sensoridataa, viestii ohjauslaitteiden ja muiden koneiden kanssa, sekä suorittaa ohjelmakoodia ja ohjaa robotin toimintaa. (Siciliano ja Khatib, 2008, ss. 982–983)

Pendantti (engl. *teach pendant*) on liikuteltava ohjauslaite, jolla robottia voidaan ohjata suoraan kentältä käsin. Kyseessä on eräänlainen kauko-ohjain, joka liitetään yleensä suoraan robotin kontrolleriin. Nykyaikaisessa *pendantissa* on yleensä näyttö robotin tilan ja ohjelman kulun esittämistä varten, sekä useissa malleissa on myös niin sanottu kuolleenmiehenkytkin. (*ABB Robotics Operating manual: RobotStudio 5.13*, 2010, s. 15; Service Point Kuopio Oy, 2014, s. 10) Tuotantotilassa olevan robotin toimintoja hallitaan usein valvomosta käsin, hyödyntäen tietokoneita ja valvomo-ohjelmistoa, mutta kuvan 2 kaltainen pendantti on silti tärkeä työkalu virhetilanteiden hallinnassa, sekä robottia ohjelmoitaessa ja käyttöönotettaessa.



Kuva 2. Pendantti. (Jul Julkiflie, *GrabCAD Community Library*, 2021)

Valvomo-ohjelmisto, eli SCADA (engl. *Supervisory Control And Data Acquisition*) on korkeamman tason ohjelmisto, jolla robotin ja tuotantolinjan toimintoja voidaan ohjata. Useimmiten SCADA huolehtii robotin ylemmän tason tehtävistä, kuten työvaiheesta. SCADA on ohjelmisto, jota robotin operaattori, eli robotin toimintaa ohjaava ja valvova henkilö hyödyntää työssään.

Robotin ohjausjärjestelmä hyödyntää antureilta saatua dataa. Sensori tai anturi on elektroninen komponentti tai laite, joka tuottaa dataa manipulaattorin, manipuloitavan objektin ja robotin ympäristön tilasta. Sensoreita on monenlaisia, aina mekaanisista kosketussensoreista konenäköä hyödyntäviin kameroihin. (Siciliano ja Khatib, 2008, s. 90)

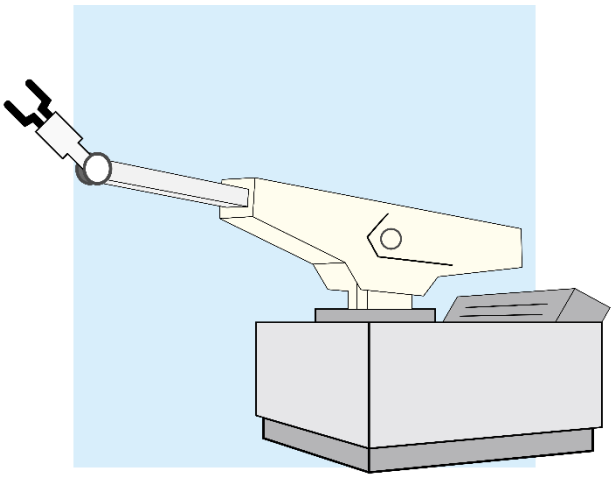
3.1.3 Robottityypit

Robotin mekaaninen tyyppi vaikuttaa suuresti robotin ominaisuuksiin, kuten toiminta-alueen muotoon, voimaan, nopeuteen, tarkkuuteen ja toistettavuuteen, sekä matemaattiseen mallintamiseen. Sopivan robottityypin valinta on riippuvainen sovelluskohteen vaatimuksista, kuten kappaleenkäsittelyrobotin liikuttamien objektien massasta. (Legnani ja Fassi, 2012, s. 10)

Suorakulmaisessa, eli karteesisessa robotissa (engl. *cartesian coordinate robot* mm. *Gantry* eli *portaalirobotit*), on kolme lineaarista liikettä, joiden suunnat (x, y, z, yhteensä 3-DOF) vastaavat kolmiulotteisen karteesisen koordinaatioton akseleita. Tallainen robotti soveltuu käytettäväksi esimerkiksi erilaisissa työstölaitteissa, sillä työkalun sijainnin hallinta on tarkkaa. (Legnani ja Fassi, 2012, s. 5) Kyseistä robottityyppiä käytetään myös erilaisissa varastoratkaisuissa, ja portaalirobotin työkalun tilalle voidaan asentaa pienempi manipulaattori. Kuvassa 3 havainnollistetun suorakulmaisen robotin ohjaaminen on intuitiivista, sillä se toimii tutussa karteesisessa koordinaatistossa.

Kuva 3. Suorakulmainen portaalirobotti. (Kareem Ahmed, *GrabCAD Community Library*, 2021)

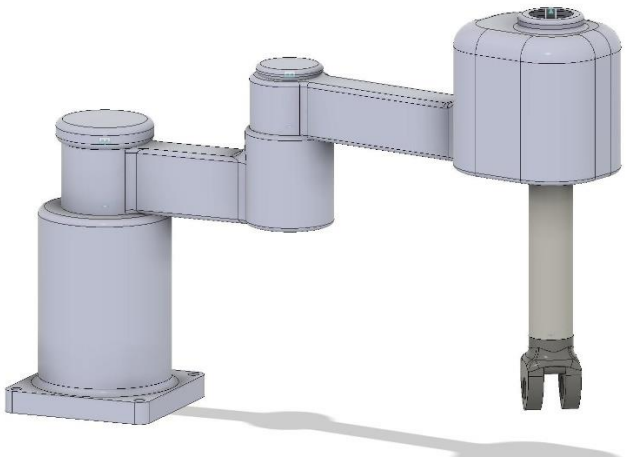
Napakoordinaatistorobotissa (engl. *polar robot* tai *spherical coordinate robot*) on yksi lineaarisesti liikkuva varsi, jota voidaan pyörittää horisontaalisesti. Myös varren korkeuskulmaa voidaan yleensä säätää, mikä on robotin kolmas liike. (Legnani ja Fassi, 2012, s. 6) On huomionarvoista, että maailman ensimmäinen teollisuusrobotti, eli kuvassa 4 esiintyvä Unimate, perustui napakoordinaatistoon (Siciliano ja Khatib, 2008, s. 964).



Kuva 4. Unimate-napakoordinaatistorobotti. (EBatlleP, *Wikimedia Commons*, 2021)

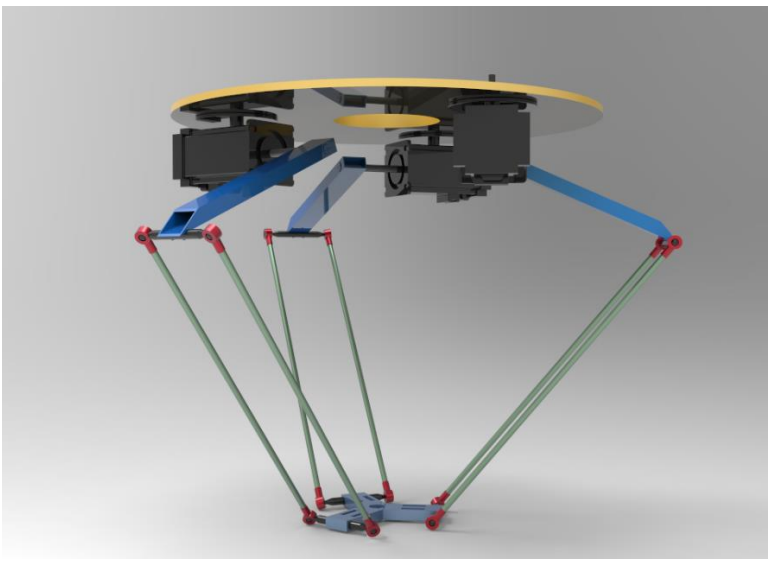
Sylinterirobotti (engl. *cylindrical robot*) on napakoodinaattirobotin monimutkaisempi muunnos. Siinä on erillinen lineaarinen liike korkeuden säätämiseen, jolloin robotin tilaan vaikuttavia muuttujia on yhteensä neljä kappaletta. (Legnani ja Fassi, 2012, s. 6)

Selective Compliance Assembly Robot, eli SCARA on monimutkaisempi versio sylinterirobotista. Kuten kuvasta 5 on nähtävissä, siinä on useita vaakatasossa pyöriviä niveliä, joiden avulla se kykenee liikuttamaan työkaluaan tasossa, sekä lineaarinen liike korkeuden säätämiseen. SCARA on erittäin tarkka ja nopea robotti muun muassa pakkaamiseen ja kappaleenkäsittelyyn, ja niitä hyödynnetään erityisesti elektroniikkateollisuudessa. (Legnani ja Fassi, 2012, s. 7)



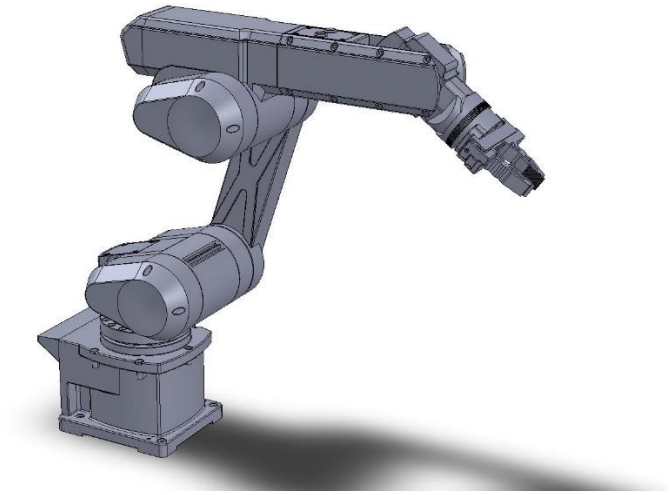
Kuva 5. SCARA robot. (Suliman Badour, *GrabCAD Community Library*, 2021)

Rinnakkaisrakenteisessa robotissa (engl. *parallel*) työkalun kolmiulotteista sijaintia hallitaan useiden rinnakkaisten nivelten avulla, aivan kuten kuvan 6 deltarobotissa. Deltarobottien hallinta on mutkikasta, ja liikealue on useimmiten suppea, mutta ne soveltuvat erinomaisesti nopeutta ja tarkkuutta vaativiin kappaleenkäsittelyn sovelluksiin. Yhteen liikesykliin voi kulua vain 0,3–0,5 sekuntia, mutta liikuteltavien objektien massat jäävät yleensä pieniksi. (Legnani ja Fassi, 2012, s. 8)



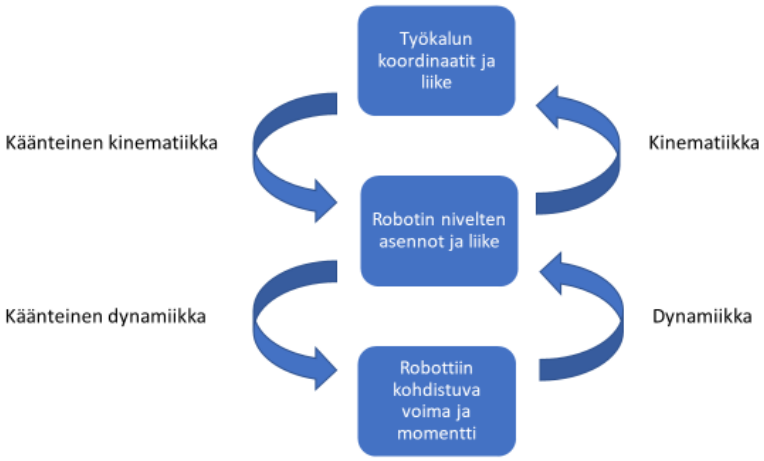
Kuva 6. Rinnakkaisrakenteinen deltarobotti. (HIGH KONCEPT, *GrabCAD Community Library*, 2021)

Kiertyvänivelinen robotti (engl. *articulated robot* tai *anthropomorphic robot*) matkii ihmisen käsien toimintaa. Useimpien mielikuva teollisuusrobotista vastaa kuvassa 7 esiintyvää nivelrobottia, sillä ne ovat erittäin yleisiä. Niissä on useimmiten 5–6 liikettä, joiden avulla ne kykenevät menestymään lähes kaikenlaisissa tehtävissä, mutta tarkka ohjaaminen vaatii suhteellisen monimutkaista matematiikkaa. (Legnani ja Fassi, 2012, s. 9)



Kuva 7. Kiertyvänivelinen teollisuusrobotti. (Kareem Ahmed, *GrabCAD Community Library*, 2021)

3.2 Matematiikka –Robotin mallinnus ja hallinta



Kuva 8. Kaavio robotin hallitsemiseen tarvittavasta matematiikasta.

3.2.1 Kinematiikka

Portaalirobotin ohjaaminen karteesisiin koordinaatteihin *x, y, z* on varsin yksinkertaista: Liikutetaan jokainen liike annettuun asemaan. Mutta mitä jos robotin liikkeiden suunnat eivät vastaa karteesista koordinaatistoa? Kuvitellaan esimerkiksi tapaus, jossa napakoordinaatistorobottia halutaan hallita karteesisten koordinaattien avulla. Työkalun sijainnin (TCP) liikuttaminen yhteen annettuun koordinaattiin voi vaatia useiden liikkeiden yhteistyötä, joten robotin hallinta on monimutkaisempaa.

Kinematiikka tutkii robotin työkalun sijainnin ja liikkeen vastetta manipulaattorin liikkuvien mekanismien, kuten robotin nivelten liikkeeseen. (Murray *ym.*, 1994, s. 81) Jacobin matriisin avulla voidaan myös löytää yhteys nivelten momenttien ja työkalun kappaleeseen kohdistaman voiman välille (Murray *ym.*, 1994, ss. 115–122), mutta saatu malli ei huomio järjestelmän dynamiikkaa, kuten manipulaattorin massan luomaa inertiaa.

Robotin vapausasteilla (engl. *Degrees Of Freedom*, DOF) kuvataan robotin liikkeitä kuvaavien riippumattomien muuttujien määrää. Käytännössä vapausasteet kuvaavat siis niitä muuttujia, joita voidaan vapaasti muuttaa, jotta esimerkiksi työkalun sijainti saadaan haluttuihin koordinaatteihin. Esimerkiksi lineaarisen toimilaitteen asema vastaa yhtä vapausastetta, mutta ihmisen olkapään kaltaisella pallonivelellä voi olla useampia vapausasteita. (Murray *ym.*, 1994, s. 84; Siciliano ja Khatib, 2008, s. 1311)

Kinemaattisessa mallinnuksessa on kyse koordinaattimuunnoksesta. Kuinka robotin vapausasteiden virittämässä avaruudessa ilmaistu manipulaattorin tila vaikuttaa karteesisessa koordinaatistossa ilmaistuun työkalun sijaintiin? Sekä työkalun liike että sijainti voidaan laskea robotin nivelten asennon ja kulmanopeuden funktiona, hyödyntämällä yksinkertaista trigonometriaa. Matemaattisessa mielessä kyse on kuvauksesta, ja käytännön toteutukset nojaavat yleensä matriisialgebraan.

Käänteisen kinematiikan avulla tietokone kykenee laskemaan, miten toimilaitteita tulee hallita, jotta työkalun sijainti (engl. *Tool Center Point*, TCP) saadaan liikkumaan halutulla nopeudella haluttuihin koordinaatteihin.

Kinemaattisilla yhtälöillä on vain yksi ratkaisu, mutta käänteisellä kinemaattisilla yhtälöillä voi olla useampia ratkaisuja. (Murray *ym.*, 1994, s. 97) Otetaan esimerkiksi kuvitteellinen tilanne, jossa siirrät kahvikuppia työpöydällä juuri tiettyä kahvikupin liikerataa hyödyntäen. Voit toteuttaa toimenpiteen kätesi nivelillä, pelkästään keskivartaloa liikuttamalla, tai pelkästään olkapään pallonivelen avulla. Kaikkien nivelten tietty tila johtaa aina tiettyyn kahvikupin paikkaan, mutta juuri tietty kahvikupin paikka voidaan savuttaa useilla eri nivelten tiloilla.

3.2.2 Singulariteetit

Robotin singulariteetille on hankala löytää yleispätevää määritelmää, sillä useimmissa kirjallisuuden lähteissä käsitellään vain yksittäisiä singulariteettityyppejä, tai niitä käsitellään vain jostakin tietystä näkökulmasta. Kirjallisuuskatsauksen perusteella voidaan kuitenkin todeta, että robotin singulaarisella konfiguraatiolla (engl. *singular configuration*) tarkoitetaan sellaista manipulaattorin tilaa, joissa robotin toimilaitteet eivät kykene vaikuttamaan robotin työkalun asemaan, liikkeeseen tai voimaan vaaditusti. (Murray *ym.*, 1994; *Six-Axis Robot Singularities*, 2007; Siciliano ja Khatib, 2008; Legnani ja Fassi, 2012)

Singulariteetti siis saavutetaan, kun robotti on jollakin tapaa kuroutunut sellaiseen asentoon, että vaaditun liikkeen suorittaminen vaatii toimilaitteilta isompaa vääntöä, voimaa, kiihtyvyyttä, nopeutta tai liikealuetta, kuin mihin robotti fyysisesti kykenee.

Palataan kahvikuppiesimerkkiin, jossa tehtävänäsi on liikuttaa työpöydälläsi olevaa kahvikuppia tiettyä rataa pitkin pisteestä A pisteeseen B. Voit siirtää kupin normaalisti, tai sitten voit pyöräyttää kyynärpääsi kohti taivasta, ja yrittää liikuttaa kahvikuppia kyynärpää ylöspäin taivutettuna. Näin toimimalla törmäät todennäköisemmin singulariteettiin, kuin kuppia luonnollisesti liikuttamalla. Voi siis olla, että vain jotkin mahdolliset työkalun liikkeen toteutukset johtavat ongelmiin singulariteettien kanssa. Kahvikupin siirtäminen on helppoa, kun hyödynnetään ihmiselle luontaista liikerataa, mutta helppous on suhteellista. Miten tietokone löytää ”luontaisimman” liikeradan?

Matemaattisesti singulariteetti ilmenee siten, että robotin hallintaan käytettyihin matriisiyhtälöihin ei löydy järkevää ratkaisua, eli jokin laskennassa käytetty matriisi menee singulaariseksi. Singulariteetin läheisyys voi näkyä siten, että robotin hallintaan käytetyt säätöarvot saavat fysikaalisesti mahdottomia arvoja. Esimerkiksi niveliltä vaaditut kulmanopeudet voivat kasvaa äärettömän suuriksi. Singulariteetti tai sen läheisyys voi johtaa laskentavirheeseen ja robotin yllättävään pysähtymiseen, tai sitten tarkkuus-, voima- ja nopeusvaatimuksiin ei yksinkertaisesti päästä. Singulariteetit voidaan jakaa erilaisiin alaryhmiin, riippuen esimerkiksi robottityypistä.

Esimerkiksi kinemaattinen singulaarinen konfiguraatio (*singular configuration*) ilmenee matemaattisesti siten, että kinemaattisen mallin hyödyntämä Jacobin matriisi menee singulaariseksi, eli se ei käänny, mikä voi johtaa robotin pysäyttävään laskennalliseen virheeseen. (Murray *ym.*, 1994, s. 124; Siciliano ja Khatib, 2008, s. 248; Legnani ja Fassi, 2012, ss. 41–44)

Robotin työkalun liikkeen toteuttamiseen on yleensä olemassa useita vaihtoehtoisia tapoja, mikäli vapausasteita on tarpeeksi paljon. Käänteisen kinemaattisen yhtälön ratkaisujen lukumäärää voidaan analysoida Jacobin matriisin ominaisuuksia tutkimalla. Matriisin aste (*Rank*) kuvaa mahdollisten ratkaisujen lukumäärää, eli matriisin aste tippuu singulariteetin läheisyydessä. (Murray *ym.*, 1994, s. 124) Tämä ei kuitenkaan ole riittävä työkalu ongelmien kiertämiseksi, sillä robotin niveliltä vaaditut kulmanopeudet ja momentit voivat kasvaa äärimmäisen suuriksi singulariteettien läheisyydessä. Manipuloitavuus (engl. *manipulability*) mittaa robotin tilan etäisyyttä singulariteeteista, eli se kuvaa robotin kykyä *saavuttaa* ja *muuttaa* erilaisia manipulaattorin asentoja, ja se on siten selvästi parempi työkalu singulariteettien kiertämiseen. (Murray *ym.*, 1994, s. 128)

3.2.3 Dynamiikka ja säätötekniikka

Robotin dynamiikka kuvaa robotin varsien liikkeiden vastetta toimilaitteiden robottiin kohdistamiin voimiin. Toimilaitteiden voidaan olettaa olevan objekteja, jotka kohdistavat robotin varsiin voimia ja momentteja, sekä kykenevät liikkumaan tietyllä tavalla. Robotin dynamiikkaa tutkittaessa ei yleensä oteta kantaa toimilaitteiden sisäiseen dynamiikkaan. (Murray *ym.*, 1994, s. 155)

Robotin dynamiikan ymmärtäminen on tärkeää, jotta manipulaattorin varsia voidaan liikuttaa hallitusti ja tarkasti. Dynamiikka on keskeinen osa-alue robotin liikkeiden säädössä ja hallinnassa. Robotic Manipulation -kirjassa robotin dynamiikkaa kuvataan joukolla epälineaarisia toisen asteen differentiaaliyhtälöitä, jotka ovat riippuvaisia robotin kinematiikasta ja inertiasta. (Murray *ym.*, 1994, s. 156)

Robotin rungon dynamiikan ymmärtäminen ei kuitenkaan riitä, sillä yleensä robotit vuorovaikuttavat ympäristönsä kanssa, eli ympäristö kohdistaa robottiin voimia. Käytännössä tämä voi tarkoittaa esimerkiksi sitä, että kappaletavararobotin poimimalla objektilla on inertia ja massa, joiden vaikutus robotin dynamiikkaan tulee pyrkiä huomioimaan. (Murray *ym.*, 1994, s. 156)

Robotin dynaamisen mallin avulla voidaan laskea, miten toimilaitteita tulee ohjata, jotta robotin varret saadaan liikkumaan halutulla tavalla, tai pysymään halutussa konfiguraatiossa. Täydellinen matemaattinen malli tarkoittaisi siis sitä, että robottia voitaisiin ohjata täysin ilman reaaliaikaista tietoa robotin tilasta, kun alkutila tunnetaan. Reaalimaailman mallit eivät kuitenkaan ole täydellisiä, joten joudumme turvautumaan takaisinkytkettyyn säätöön.

Takaisinkytketty, suljettu säätöpiiri hyödyntää sensoridataa säädön apuna. Nyt säätötoimenpide on riippuvainen halutun ja havaitun arvon erotuksesta, mutta optimaalisen säätötoimenpiteen löytäminen perustuu edelleen robotin dynamiikan tuntemukseen. (Murray *ym.*, 1994, s. 156)

3.3 Ohjelmisto –Tietokone ohjaa robottia

Perinteinen teollisuusrobottiohjelma on joukko liikesarjasekvenssejä, jotka koostuvat koordinaattipisteistä, ja niiden välillä siirtymiseen käytetyistä liikeradoista. Robotin noudattamat liikeradat ovat siis ennalta ohjelmoituja, eikä robotti osaa mukauttaa liikerataansa sensoreilta saamansa informaation avulla. (Legnani ja Fassi, 2012, s. 15)

Robotit ovat kuitenkin muuttumassa entistä älykkäämmiksi, sillä yhä useammat robotit kykenevät hyödyntämään sensoridataa liikkeidensä suunnittelussa ja hallinnassa. Liikeradat voidaan generoida automaattisesti online-tilassa (*live programming*), perustuen esimerkiksi konenäköön ja syvyyskameraan. Otetaan esimerkiksi hiontarobotti, jonka tehtävänä on liikuttaa hiontalaikkaa kappaleen pinnanmuotoja mukaillen. Pienikin heitto kappaleen ennaltaohjelmoidussa muodossa voi pilata hionnan, jos robotti ei kykene mukauttamaan liikkeitänsä sensoridatan avulla. (Legnani ja Fassi, 2012, s. 15)

Älykkäimmät teollisuusrobotit ovat kykeneviä oppimaan, ja kehittymään työssään itsenäisesti (Legnani ja Fassi, 2012, s. 15). Lisäksi uudet älykkäät ohjelmointimenetelmät helpottavat ja automatisoivat myös perinteisten robottiohjelmien tekemistä, mikä nopeuttaa ohjelmointia, ja pienentää seisokkien määrää sekä kestoa.

Robotin ohjelmoijan ei tavallisesti tarvitse hallita kinematiikkaa ja dynamiikkaa, sillä ohjelmoija voi hyödyntää robottivalmistajan tai ohjelmointiympäristön tarjoamia valmiita funktioita ja aliohjelmia. Kuten ohjelmoitaessa yleisesti, koodaajan ei tarvitse ymmärtää kaikkien käyttämiensä funktioiden syvällistä toimintaa.

Robotteja ohjelmoidaan yleensä työkalun sijainnin ja asennon avulla, eli ohjelmassa hyödynnetään TCP-koordinaatteja, joiden pohjalta nivelten asento lasketaan käänteisen kinematiikan avulla. Yleensä liikeradan ohjelmointi onnistuu vain muutaman pisteen turvin, sillä robotti osaa siirtyä pisteiden välillä tiettyä geometriaa mukailen, esimerkiksi, optimaalisesti, kaarevasti, tai lineaarisesti suoraan. (Legnani ja Fassi, 2012, s. 14)

Lineaarinen, eli viivasuora siirtymä on nivelrobotin kannalta haastava, ja usein myös hidas vaihtoehto, sillä täydellisen lineaarinen siirtymä vaatii useiden nivelten välistä yhteistyötä. Täysin lineaarinen liikerata vaatii kovaa keskittymistä jopa ihmiseltä, kuten voit kahvikuppia siirtämällä havaita. Esimerkiksi nopeuden kannalta optimaalisin liikerata on vain harvoin lineaarinen, jos käytössä on nivelrobotti. Useimmissa ohjelmointityökaluissa onkin toiminto, joka mahdollista robotin toiminnan kannalta optimaalisen polun automaattisen hyödyntämisen.

3.3.1 Ohjelmointikielet

Perinteiset robottien ohjelmointikielet eroavat jonkun verran yleiskielistä. Niissä on muutamia erikoipiirteitä, jotka helpottavat robotin ohjelmointia. Useimmissa robottikielissä on esimerkiksi mahdollisuus takaperinsuorittamiseen, mikä mahdollistaa ohjelmoidun sekvenssin suorittamisen takaperin. (Siciliano ja Khatib, 2008, ss. 976–979) Tämä on hyödyllinen ominaisuus esimerkiksi silloin, kun robotin työkalu halutaan ensin liikuttaa johonkin pisteeseen, ja sitten takaisin, tai robotti törmää ei-halutusti johonkin kappaleeseen.

Useimmat robottikielet ovat interaktiivisia, eli ohjelman kulkua voidaan seurata, ja jopa muokata reaaliajassa (Siciliano ja Khatib, 2008, ss. 976–979). Esimerkiksi pendantti osaa yleensä esittää, missä kohti ohjelmaa ollaan menossa. Ohjelmoijan ei siis tarvitse kiinnittää huomiota erilliseen käyttäjän kanssa viestimiseen, jotta ohjelman suoritusvaiheiden kulkua voidaan seurata.

Robottien ohjelmoimiseen on kehitetty vuosien saatossa useita kieliä, jotka perustuvat ainakin löyhästi johonkin yleiskieleen. Lähes jokaisella isolla robottivalmistajalla on oma ohjelmointikielensä. (Legnani ja Fassi, 2012, s. 19) Alla on listattuna muutamia tunnetuimpia kieliä:

- RAPID (ABB)
- KRL (KUKA)
- Karel (Fanuc)
- SLIM (Jis8436§)
- LRP (open source)

Useimpien kielten syntaksit muistuttavat toisiaan, mutta erilliset robottikielet vaativat ohjelmoijalta perehtyneisyyttä spesifiseen, ja mahdollisesti myös hyödynnettävyydeltään rajoittuneeseen ohjelmointikieleen. Spesifinen robottikieli ei välttämättä sovellu älykkäiden robottien ohjelmoimiseen, sillä tuetut toiminnot rajoittuvat usein valmistajakohtaisiin ratkaisuihin.

Spesifisen robottikielen osaamisvaatimus hankaloittaa myös tehtävään soveltuvien henkilöiden löytämistä, ja erillisten koulutusten järjestäminen maksaa, sekä järjestelmän ylläpidettävyys kärsii. Nykyaikaisten, syntaksiltaan ”löysien”, korkeamman tason kielten käyttö onkin ehkä yleistymään päin. Esimerkiksi Python mahdollistaa ainakin useimpien robottikielistä tuttujen toimintojen toteuttamisen, ja se on ominaisuuksiltaan huomattavasti monipuolisempi älykkäitä robotteja ohjelmoitaessa.

3.3.2 Online-ohjelmointi

Online-ohjelmoinnilla tarkoitetaan niitä *in situ* -menemiä, joissa itse fyysistä robottia käytetään ohjelmoinnin ja *debuggauksen* apuna (Siciliano ja Khatib, 2008, s. 977). On ilmeistä, että ohjelmoinnin apuna toimiva robotti ei voi samanaikaisesti toimia tuotannossa, mikä selkeä ongelma. Online-ohjelmointi on kuitenkin ollut pitkään vallitseva ohjelmointimenetelmä, sillä se on yksinkertainen, eikä se vaadi monimutkaisia simulaattoreita.

Online-ohjelmointi voi yksinkertaisimmillaan tarkoittaa sitä, että koodari istuu robotin vieressä, ja kirjoittaa perinteistä koodia, jota voidaan testata heti. Ensiyrittämältä toimivan koodin kirjoittaminen on vaikeaa. On siis tärkeää, että ohjelmoija pääsee testaamaan ohjelmansa osia mahdollisimman nopeasti ja helposti, joko fyysisessä, tai simuloidussa ympäristössä.

Nykyisin robotteja ohjelmoidaan yleensä opettamalla, eli operaattori ohjaa, eli *joggaa* robotin *pendantin* avulla eri asentoihin, jotka tallennetaan, ja joiden välillä robotti kykenee liikkumaan automaattisesti opetettua polkua pitkin. (Pan, Polden, *ym.*, 2012, s. 88)

Useampiakselisen robotin *joggaaminen* on kuitenkin suhteellisen hankala tehtävä. Monimutkaisten liikkeiden ohjelmoiminen *joggaamalla* vaatii ohjelmoijalta ammattitaitoa ja kokemusta, joten on kehitetty robotteja, joita voidaan opettaa helpommin fyysisen kontaktin kautta. Robotti kykenee tunnistamaan ihmisen siihen kohdistaman voiman, ja mukailemaan voiman osoittamia liikkeitä. Käytännössä ihminen siis ottaa robotista kiinni, löysää mahdollisen jarrun ja vääntää robotin haluttuun asentoon, jolloin robotti tallentaa näytetyt sijainnit ja muodostaa niiden välillä liikkumiseen tarvittavat liikeradat. (Pan *ym.*, 2012, s. 88)

3.3.3 Offline-ohjelmointi (OLP)

Offline-ohjelmointi, eli OLP eroaa online-ohjelmoinnista siten, ettei ohjelmoinnin avuksi tarvita fyysistä tuotannon robottia. Tietotekniikan kehittyminen on mahdollistanut sen, että robotin toimintaa voidaan simuloida tietokoneella. Näin fyysistä robottia ja sen ympäristöä ei enää tarvita ohjelmoinnin avuksi, sillä ne kyetään muodostamaan virtuaalisesti. Offline-ohjelmoinnin ansiosta robottia voidaan opettaa ja ohjelmoida tehokkaammin, tarkemmin ja nopeammin, eikä tuotantoa tarvitse pysäyttää ohjelmoinnin ajaksi. (Pan *ym.*, 2012, s. 90)

Robotin liikeradat voidaan myös generoida suoraan 3D-mallista. On tunnettua, että esimerkiksi CNC jyrsimet, tulostimet ja ladontakoneet hyödyntävät tallaista lähestymistapaa, jossa robotin liikkeet generoidaan suoraan ”piirustuksista”. Vastaava idea toimii myös muunlaisia robotteja ohjelmoitaessa, kun hyödynnetään älykkäitä algoritmeja.

Useimmista simulointiympäristöistä löytyykin toiminto, jolla robotin liikeradat saadaan mukailemaan CAD-mallin pintaa, mistä hyötyä esimerkiksi maalaus, hitsaus ja hiontarobottien ohjelmoinnissa. Esimerkiksi VisualComponents simulointiohjelmassa kyseinen toiminto tunnetaan nimellä *Curve-Teaching-Tool*. (Visual Components, 2017)

Simulointi ei välttämättä vaadi erillisten kolmiulotteisten mallien piirtämistä, sillä koneiden suunnittelussa hyödynnettäviä piirustuksia voidaan useimmiten hyödyntää sellaisenaan simuloitujen objektien luomiseen. Robotin, tuotantotilan ja käsiteltävien objektien mallintamiseen voi kuitenkin kulua merkittävästi aikaa ja rahaa, jos valmista 3D-mallia ei ole saatavilla, mutta 3D-skannausmenetelmät helpottavat mallintamista.

Tarvittavat ohjelmistot voivat kuitenkin olla kalliita, ja useat niistä ovat myös valitettavan monimutkaisia käyttää, joten online-menetelmät ovat edelleen yleisiä erityisesti pienen mittakaavan tuotannossa. (Pan *ym.*, 2012, s. 90)

3.3.4 Simulointiympäristöt

Simulointiohjelmistot mahdollistavat fyysisen maailman objektien ja ilmiöiden reaaliaikaisen mallinnuksen. Simulointiympäristöt sisältävät fysiikkamoottorin, jolla ne kykenevät simuloimaan reaalimaailman fysiikkaa, kuten objektien massaa, inertiaa, painovoimaa ja kimmoisuutta. Näin tuotantolaitteiden ja robottien ominaisuuksia, kuten robottien kinematiikkaa ja dynamiikkaa voidaan simuloida tehokkaasti, mikä mahdollistaa offline-ohjelmoinnin.

Robottien, tuotantosolujen, tuotantolinjojen, operaattoreiden, ja jopa heidän hyödyntämien käyttöliittymien toimintaa voidaan tutkia ja kehittää yhden ainoan, kolmiulotteisen ja reaaliaikaisen simulaation avulla. (*esimerkiksi ABB Robot Studio*)

Simulointi on työkalu, joka helpottaa ja tehostaa suunnittelua. Virheet, vaaranpaikat ja muut ongelmat voidaan havaita hyvissä ajoin, jopa ennen tehtaan perustusten muuraamista. Fyysisesti rakennettujen asioiden korjaaminen ja muokkaaminen on kallista, joten ongelmien aikainen havaitseminen ja optimoiminen säästää sekä aikaa että rahaa.

Virtuaalilasit (*Virtual Reality*, VR) avaavat uuden, entistä realistisemman tavan simuloinnin tarkasteluun. Suunnittelijat ja ohjelmoijat pääsevät tarkastelemaan simulaatiota entistä luonnollisemmin, mikä helpottaa työtä. Virtuaalilasien ja ohjainten avulla robotteja voi opettaa virtuaalisessa online-tilassa, jossa ihminen pystyy vuorovaikuttamaan robottien kanssa virtuaalisesti. (Visual Components, 2017)

Lisätyn todellisuuden teknologiat (engl. *Mixed Reality* eli MR ja *Augmented Reality* eli AR) mahdollistavat simuloinnin yhdistämisen online-ohjelmointiin, mikä hämärtää online- ja offline-ohjelmoinnin rajaa entisestään. Esimerkiksi simuloinnin tuloksia on mahdollista tarkastella AR-laseja hyödyntäen, myös *in situ*. (Neves, Serrario, ym., 2018)

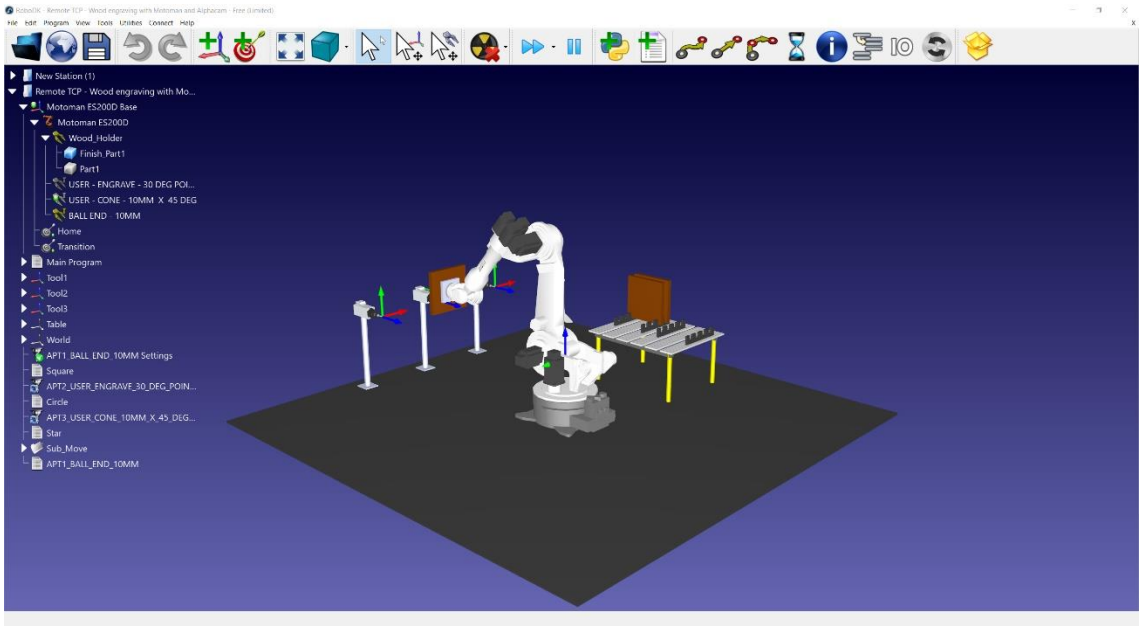
Robottien simuloimiseen ja offline-ohjelmointiin on kehitetty useita ohjelmistoja, joista mainittakoon ainakin seuraavat:

- ABB RobotStudio
- RoboDK
- Webots
- Gazebo
- OpenRAVE
- Visual Components
- RoboLogix
- Delmia

ABB RobotStudio on työkalu, jolla robotteja voi ohjelmoida graafisten työkalujen ja simuloinnin avulla, ilman tarvetta tekstipohjaisen koodin kirjoittamiselle. Graafisten työkalujen avulla generoidaan RABIT-koodia, jota ABB:n robottien ohjaimet ymmärtävät, ja jota ohjelmoija voi myös halutessaan muokata manuaalisesti. Simulaatio on melko autenttinen, sillä simuloitujen ohjausyksiköiden toiminta perustuu samaan koodiin, jota oikeat fyysiset ABB:n robottiohjaimet hyödyntävät. (*ABB Robotics Operating manual: RobotStudio 5.13*, 2010) RobotStudiossa on myös rajapinta Microsoft Visual Studio -ohjelmistolla tehdyille applikaatioille, mikä mahdollistaa joidenkin tunnettujen yleiskielten hyödyntämisen robottia ohjelmoitaessa (Connolly, 2009, s. 541).

Visual Components on suomessa kehitetty simulointiohjelma, joka soveltuu tuotannon kokonaisvaltaiseen simulointiin. Ohjelmisto on arkkitehtuuriltaan modulaarinen ja perustuu osin avoimeen lähdekoodiin. Samalla ohjelmalla voidaan ohjelmoida useiden eri robottivalmistajien robotteja, ja simuloida niiden toimintaa tehokkaasti. Robottiohjelmointi onnistuu esimerkiksi graafisten työkalujen ja Python-ohjelmointikielen avulla. (Visual Components, 2017)

RoboDK robottien ohjelmoimiseen erikoistunut simulointiohjelmisto, joka tukee useiden robottivalmistajien robotteja. Myös RoboDK kykenee simuloimaan kokonaisten tuotantosolujen toimintaa, ja mukana on myös fysiikkamallinnusta. Monipuolisten rajapintojen ansiosta RoboDK tukee useita tunnettuja ohjelmointikieliä, ja ohjelmointi onnistuu myös graafisten työkalujen avulla. (RoboDK, 2021)



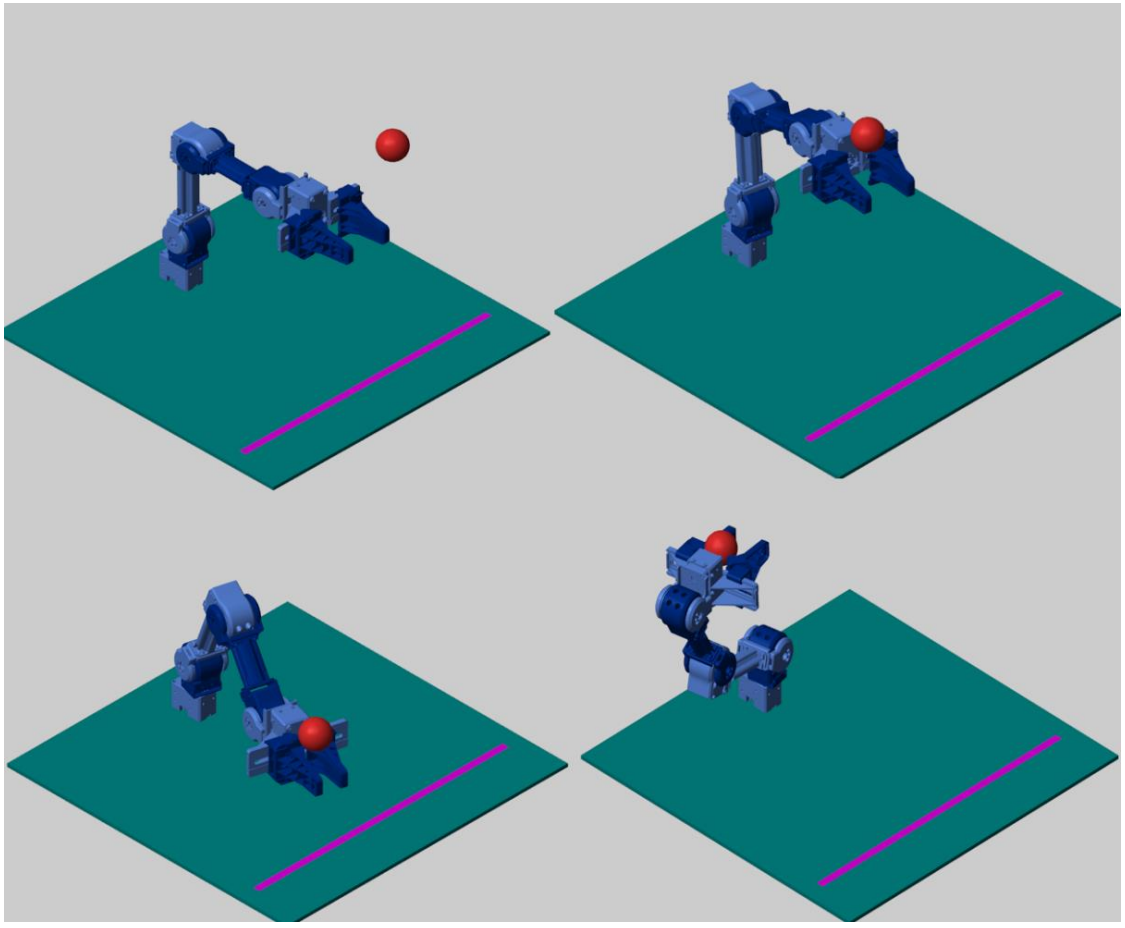
Kuva 9. Kuvankaappaus RoboDK 5.2 simulointiohjelmistosta.

Lyhyen kokeilun perusteella RoboDK:n käyttöliittymä on melko selkeä, kuten kuvasta 9 on havaittavissa. Ohjelma tukee useita yleiskieliä, joita ovat Python, Matlab, C, C#, C++ ja Visual Basic. RoboDK kääntää yleiskielisen ohjelman jollekin lukuisista tuetuista robottikielistä, joten lopullinen robotille ladattava ohjelma on yleensä vain kokoelma koordinaatteja, ja niiden välillä liikkumiseen käytettäviä käskyjä. Yleiskielellinen ohjelma voi kuitenkin hyödyntää jotain monimutkaisempaa algoritmia, tai jopa keinoälyä, jolla robottikielisen ohjelman hyödyntämät pisteet generoidaan.

MATLAB Robotic Toolbox sisältää runsaasti työkaluja robottien kinematiikan, dynamiikan ja liikeratojen suunnittelemiseen ja simulointiin, sekä törmäysten havaitsemiseen. MATLAB SimScape mahdollistaa robotin kolmiulotteisen mallintamisen, ja tarvittaessa Robotic Toolbox yhdistyy myös suoraan Gazedo-simulointiohjelmistoon. (MathWorks, 2021)

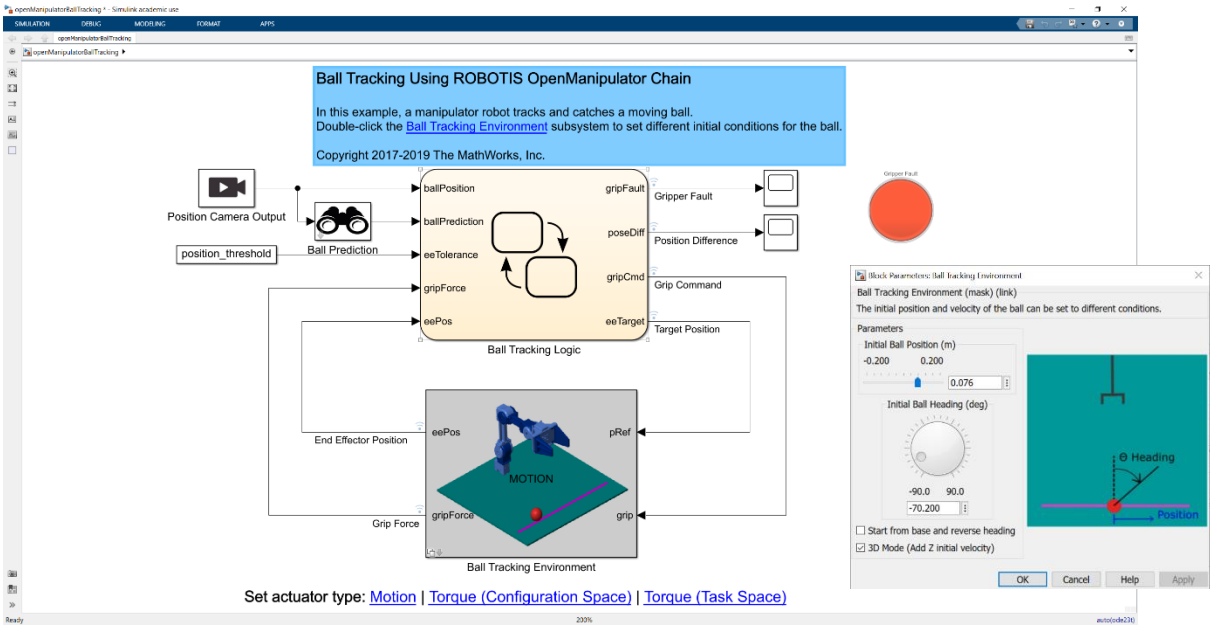
MATLAB osaa myös kommunikoida ROS-yhteensopivien robottien kanssa, ja sen avulla suunniteltujen liikesarjojen perusteella on myös mahdollista generoida robottikoodia, kun hyödynnetään esimerkiksi RoboDK-simulointiohjelmistoa.

GitHub-palvelusta on myös saatavilla demoja, joita MATLAB-lisenssin haltiat voivat kokeilla vapaasti. Kuva 10 esittää eräästä kokeilemastani simulaatiosta (*Robot Manipulator Control Example*, 2019) napattua kuvasarjaa, jossa älykäs robotti pyrkii nappaamaan ilmaan heitetyn pallon. On huomionarvoista, että myös pallon fysikkaa simuloidaan reaaliaikaisesti. Robotti kykenee optimoimaan liikkeensä satunnaisesti heitetyn pallon liikeradan mukaisesti, ja napaamaan pallon kiinni.



Kuva 10. Sarja kuvankaappauksia MATLAB-pohjaisesta robottisimulaatiosta.

Simulaatio toimii hyvin, ja yleensä robotti onnistuu tehtävässään, mutta tietyt lentoradan parametrit tuottavat vaikeuksia. Kuvassa 11 esitellyillä parametreillä robotti tasapainottelee palloa pihtiensä päällä kuin hylje, eikä onnistu saamaan siitä pitävää otetta, avain kuten edellä esitellystä kuvarjasta voidaan havaita.



Kuva 11. Kuvankaappaus MATLAB-pohjaisen robottisimulaation hyödyntämästä Simulink-mallista.

3.3.5 Robotic middleware –Robot Operating System (ROS)

Älykkäiden robottijärjestelmien kehittäminen on erittäin haastavaa, sillä kehitystyö voidaan jakaa lukuisiin yksittäisiin osaongelmiin, joiden ratkaiseminen on todella aikaa ja resursseja kuluttavaa. Yksittäinen taho ei kykene tekemään kaikkea, ainakaan järkevässä aikataulussa. (*ROS.org | Powering the world's robots*, 2021)

Standardien puute ja sekavuus, sekä monikäyttöisten ohjelmakomponenttien ja kirjastojen puute on ollut valtava ongelma älykkään robotiikan kehitykselle. Pyörää ei kannattaisi keksiä aina uudelleen, eivätkä edes isoimpien valmistajien resurssit riitä kaikkien osaongelmien tehokkaaseen ratkaisemiseen. Aiemmin on silti ollut valitettavan yleistä, että jokaisessa älykkään robotiikan tutkimushankkeessa on luotu omat erilliset versiot robottien käyttämistä perusalgoritmeista, ja teollisuuden sovellukset ovat nojanneet pitkälti kankeisiin robottivalmistajakohtaisiin ratkaisuihin. Onneksi 2000 luvun puolessa välissä alkunsa saanut *Robot Operating System* (ROS) on tuonut helpotusta tähän ongelmaan. (RoboAI, 2021)

Nimestään huolimatta *Robot Operating System* eli ROS ei varsinaisesti ole käyttöjärjestelmä, vaan kyseessä on niin sanottu metakäyttöjärjestelmä, joka on ennemminkin kokoelma toisensa kanssa yhteensopivia kirjastoja, käytäntöjä ja työkaluja, joilla eri ohjelmakomponentit ja laitteet kykenevät toimimaan tehokkaasti yhdessä. Kyseessä on myös avoimen lähdekoodin yhteisö, jossa älykkäiden robottien tekemiseen tarvittavaa tietotaitoa ja ohjelmistoteknologiaa jaetaan käyttäjien kesken. (Pyo, Cho, *ym.*, 2017)

ROS koostuu useista erillisistä ohjelmakomponenteista, joita kutsutaan *nodeiksi*, eli solmuiksi (engl. *node*). *Nodeja* voidaan ohjelmoida vapaasti useilla eri ohjelmointikielillä, joten ROS on kieliriippumaton. *Node* voi olla esimerkiksi aliohjelma, joka suunnittelee robotin liikkeet kameran kuvasignaalia käsittelevän noden muodostaman analyysin pohjalta. (*Nodes - ROS Wiki*, 2021)

Erilaisia *nodeja* on saatavilla vapaasti avoimen lähdekoodin periaatteita noudattaen, joten robottiohjelmoijan ei tarvitse tehdä kaikkea itse. ROS on siis loppupeleissä lähinnä kokonaisuus, joka määrittelee miten erilaisten ohjelmakomponenttien tulisi toimia keskenään, ja kuinka ohjelmakomponenttien välinen viestintä tulisi hoitaa. *Node* ei siis ole ainoa komponenttityyppi, jonka ROS määrittelee, vaikka *nodet* ovatkin keskeisessä asemassa ROS-järjestelmän toiminnan kannalta.

Vastaavia myös väliohjelmistona, eli *middlewarena* tunnettuja ohjelmistokomponentteja on olemassa muitakin (Naidoo, Bright, *ym.*, 2016, s. 985), mutta ROS on noussut suosituimmaksi älykkään robotiikan alustaksi. ROS helpottaa älykkään robotiikan kehittämistä, tulosten kaupallistamista ja uuden teknologian tuomista teollisuuskäyttöön.

ROS-mahdollistaa myös hajautetun laskennan, eli eri *nodeja* voidaan ajaa eri tietokoneilla. Mutta miten älykkään robotin implementointi ja ohjelmointi eroaa tavallisten ”tyhmien” robottien suunnittelusta ja ohjelmoinnista?

Robottien liikesarjasekvenssejä koordinoiva tietokoneohjelma pyörii perinteisesti robotin omassa ohjausyksikössä, mutta älykkäitä robotteja ohjataan yleisesti ulkopuolisen tietokoneen avulla, joka lähettää komentoja robotille. Robotin ohjausyksikön laskentateho ei yleensä riitä laskentaintensiivisiin keinoälysovelluksiin, joten ulkoinen ohjaus on välttämätöntä. Lisäksi valmistajakohtaiset ratkaisut ovat usein kankeita, eivätkä ne useinkaan mahdollista älykkäiden keinoälypohjaisten ratkaisujen kehittämistä.

Useimmat isot robottivalmistajat tukevat ROS-järjestelmää, ja ROS sisältää myös valtavan määrän ajureita erilaisille sensoreille, kuten kameroille ja LiDAR-antureille. (*ROS.org | Powering the world’s robots*, 2021) ROS on siis se linkki, joka yhdistää robottijärjestelmän komponentit toisiinsa, ilman että ohjelmoijan tarvitsee itse ohjelmoida kaikkea alusta alkaen.

Otetaan esimerkiksi aiemmin esitelty simulaatiodemo, jossa robotti pyrkii nappaamaan ilmaan heitetyn pallon kiinni. ROS tarjoaa tarvittavat ajurit, joiden avulla MATLAB-ohjelmistoa pyörittävä tietokone voi viestiä robotin, ja pallon sijainnin selvittävän syvyyskameran kanssa. Sen ansiosta kaikkia rajapintoja ei tarvitse ohjelmoida itse, mikä helpottaa simuloidun demon implementoimista käytännön sovellukseen.

4 YHTEENVETO

Robotit ovat yleistyneet kovaa tahtia viime vuosina. Nykyiset robotit soveltuvat myös sellaisiin tehtäviin, joihin teollisuusrobottien ei uskottu kykenevän vielä kymmenen vuotta sitten. Konenäköä hyödyntävä yhteistyörobotiikka on hyvä esimerkki uuden teknologian tuomista mahdollisuuksista. Ihmisen kanssa samassa tilassa yhteistyötä tekevien cobottien voidaankin olettaa yleistyvän lähitulevaisuudessa.

Robottien ohjelmoinnin ketteryyden merkitystä on hankala korostaa kylliksi. Jopa perinteisesti volyymifokusoitunut tuotanto hyötyy uusista entistä tehokkaammista ohjelmointitavoista, ja robottien kasvaneesta älykkyydestä. Erityisesti lisätyn todellisuuden teknologioilta on lupa odottaa helpotusta robottien ohjelmoimiseen ja simuloimiseen lähitulevaisuudessa.

Robottien kinematiikan ja dynamiikan perusteiden ymmärtäminen on tärkeää, jotta robottien ohjelmointi onnistuu sujuvasti. Ihmisille itsestään selvät ja helpot asiat eivät välttämättä ole itsestäänselvyyksiä koneelle. Esimerkiksi singulariteettien ymmärtäminen on tärkeää, kun tässä työssä esitellyistä perusasioista jatketaan robotin liikeratojen optimointiin, ja erilaisiin keinoälypohjaisten ohjausmenetelmien kehittämiseen.

Älykäs teollisuusrobotti voi tunnistaa kohteensa, ja generoida liikeratansa automaattisesti tilanteineen mukaan. Tällaisen robotin ohjelmoiminen ja kehittäminen on *hankala* tehtävä. Esimerkiksi liikeratojen generoimisessa tulee huomioida lukuisia asioita, ja sidosryhmiä on useita. On tärkeää ymmärtää, että kaikkea ei aina kannata tehdä itse, varsiinkaan silloin, kuin tarkoituksena on implementoida älykäs robotti oikeaan teollisuuden sovellukseen.

ROS on keskeinen työkalu älykkäiden robottien ohjelmoimiseen. ROS-ekosysteemi tarjoaa runsaan valikoiman valmiita työkaluja ja kirjastoja, mutta sen käyttöä opetetaan suomalaisissa kouluissa suhteellisen vähän. Onneksi ROS on myös yhteisö, josta saa monenlaista apua älykkään robottiratkaisun kehittämiseen. Ideana on jakaa osaamista ja ideoita maailmanlaajuisesti, mikä nopeuttaa robotiikan kehitystä.

LÄHDELUETTELO

ABB AB, 2010, *ABB Robotics Operating manual: RobotStudio 5.13* 2010. Västerås, Sweden. Saatavissa: https://library.e.abb.com/public/244a8a5c10ef8875c1257b4b0052193c/3HAC032104-001_revD_en.pdf.

Adept Technology, 2007. *Six-Axis Robot Singularities* [verkkodokumentti]. Saatavissa: https://www.yumpu.com/en/document/read/8700664/six-axis-robot-singularities-your-intelligent-robotics-partner- (Viitattu: 18. tammikuuta 2021).

Bughin J., Hazan E., Ramaswamy S., Chui M., Allas T., Dahlström P., Henke N. & Trench M., 2017. *Artificial intelligence – the next digital frontier?*. McKinsey Global Institute. Saatavissa: https://www.mckinsey.com/~/media/mckinsey/industries/advanced electronics/our insights/how artificial intelligence can deliver real value to companies/mgi-artificial-intelligence-discussion-paper.pdf.

Connolly C., 2009. ”Technology and applications of ABB RobotStudio”, *Industrial Robot*, 36 (6), ss. 540–545. doi: 10.1108/01439910910994605.

GrabCAD Community Library 2021. [verkkodokumentti]. Saatavissa: https://grabcad.com/library (Viitattu: 26. maaliskuuta 2021).

International Federation of Robotics, 2020. ”Executive Summary World Robotics 2020 Industrial Robots”, *World Robotic Report*, ss. 13–16. Saatavissa: https://ifr.org/free-downloads/.

ISO/TR 20218-1:2018, 2018. *Robotics — Safety design for industrial robot systems*. Geneva, Switzerland: International Organization for Standardizationtion. Saatavissa: https://www.iso.org/obp/ui/#iso:std:iso:tr:20218:-1:ed-1:v1:en (Viitattu: 18. tammikuuta 2021).

ISO 8373:2012, 2012. *Robotics — Robots and robotic devices.* Geneva, Switzerland: International Organization for Standardizationtion. Saatavissa: https://www.iso.org/standard/55890.html (Viitattu: 14. tammikuuta 2021).

Legnani G. & Irene Fassi (toimittajat), 2012. *Robotics : State of the Art and Future Trends*. New York: Nova Science Publishers, Inc (Computer science, technology and applications).

Maragkos C., Vosniakos G. C. & Matsas E, 2019. ”Virtual reality assisted robot programming for human collaboration”, *Procedia Manufacturing*, 38 (2019), ss. 1697–1704. doi: 10.1016/j.promfg.2020.01.109.

MathWorks, 2021. *Matlab Documentation*. [verkkodokumentti]. Saatavissa: https://in.mathworks.com/help/index.html?s_tid=CRUX_lftnav (Viitattu: 16. maaliskuuta 2021).

Murray R. M., Li Z. & Sastry S. S., 1994. *A mathematical introduction to robotic manipulation*. Boca Raton: CRC.

Naidoo N., Bright G. & Stopforth R., 2016. ”The Cooperation of Heterogeneous Mobile Robots in Manufacturing Environments using a Robotic Middleware Platform”, *IFAC-PapersOnLine*, 49 (12), ss. 984–989. doi: 10.1016/j.ifacol.2016.07.570.

Nodes - ROS Wiki 2021. [verkkodokumentti]. Saatavissa: http://wiki.ros.org/Nodes (Viitattu: 18. maaliskuuta 2021).

Pan Z. Polden J., Larking N., Stephen V. D. & Norrish J., 2012. ”Recent progress on programming methods for industrial robots”, *Robotics and Computer-Integrated Manufacturing*, 28 (2), ss. 87–94. doi: 10.1016/j.rcim.2011.08.004.

Pietikäinen M. & Silven O., 2019. *Tekoälyn haasteet: koneoppimisesta ja konenäöstä tunnetekoälyyn*. Oulu: Konenäön ja signaalianalyysin keskus. Saatavissa: http://urn.fi/urn:isbn:9789526224824.

Pyo Y., Cho H., Ryuwoon J. & Lim T., 2017. *ROS Robot Programming From The Basic Concept To Practical Programming and Robot Application*. Seoul, Republic of Korea: ROBOTIS Co.,Ltd.

RoboAI, 2021. *Ros (Robot Operating System) -täydennyskoulutuskokonaisuus - Roboai*. [verkkodokumentti]. Saatavissa: https://www.roboai.fi/taydennyskoulutus/ros/ (Viitattu: 18. maaliskuuta 2021).

RoboDK, 2021. *RoboDK*. [verkkodokumentti]. Saatavissa: https://robodk.com/.

MathWorks, Inc., 2019. *Robot Manipulator Control Example* 2019. [verkkodokumentti]. Saatavissa: https://github.com/mathworks-robotics/designing-robot-manipulator-algorithms (Viitattu: 26. maaliskuuta 2021).

ROS.org | Powering the world's robots 2021. [verkkodokumentti]. Saatavissa: https://www.ros.org/ (Viitattu: 16. maaliskuuta 2021).

Service Point Kuopio Oy, 2014. *Siirrettävän kappaleenkäsittelyrobottisolun käyttöohje*. Kuopio. Saatavissa: https://www.ysao.fi/loader.aspx?id=9a95c868-d42d-4b86-969f-7ea8fb570443.

Siciliano B. & Khatib O. (toimittajat), 2008. *Springer handbook of robotics*. 2. p. Berlin: Springer, Cham. doi: https://doi.org/10.1007/978-3-319-32552-1.

Visual Components, 2017. *Teaching Robots with Visual Components - Visual Components*. Saatavissa: https://www.visualcomponents.com/insights/articles/teaching-robots-visual-components/ (Viitattu: 15. tammikuuta 2021).

Vysocky A. & Novak P., 2016. ”Human - Robot collaboration in industry”, *MM Science Journal*, 2016-June, ss. 903–906. doi: 10.17973/MMSJ.2016_06_201611.

Wikimedia Commons, 2021. [verkkodokumentti]. Saatavissa: https://commons.wikimedia.org/wiki/Main_Page (Viitattu: 26. maaliskuuta 2021).